5

Life Cycle Cost Considerations for Complex Systems

John V. Farr
United States Military Academy
USA

1. Introduction

Because of complexity and technology, the upfront costing of complex systems has become a tremendous challenge. We understand how to cost hardware and to a lesser extent software. However, we are still developing tools and processes for costing the integration and interfaces of complex systems. As we scale to larger and more complex systems, system-of-systems (SoS), and enterprises our ability to determine costs becomes less relevant and reliable. Our estimates can be off by an order of magnitude. Unfortunately, this often the result of requirements creep as much as it is our inability to translate requirements to products.

Cost estimation techniques can be divided into three categories: parametric costs estimates or PCEs, analogies, and detailed engineering builds. Figure 1 shows their applicability throughout a typical product life cycle. We chose to ignore accounting in the chapter. However, capturing expenses in a formal manner is certainly the best way to ascertain costs. Obviously, developing true costing amounts and utilizing good cost management requires good accounting practices and the tracking of expenses using activity based costing techniques. Table 1 summarizes the advantages and disadvantges of these various techniques.

In this chapter we present some of the methods, processes, tools (MPTs) and other considerations for conducting analysis, estimation and managing the life cycle costs (LCCs) of complex systems.

2. Life cycle considerations

In today's global business environment, engineers, information technology professionals and practitioners, and other related product development professionals integrate hardware, software, people, and interfaces (i.e., complex systems) to produce economically viable and innovative applications while ensuring that all pieces of the enterprise are working together. No product or services are immune from cost, performance, schedule, quality, and risks and tradeoffs. Yet engineers spend most of their formal education focused on performance and most of their professional careers worrying about resources and schedule. Too often we become fixated on the technical performance to meet the customer's expectations without

worrying about the downstream costs that contribute to the total LCCs of a system. Unfortunately, in many cases the LCCs or total ownership costs (TOCs) are ignored because either the total costs would make the project untenable (especially for large government projects) or the increased acquisition costs needed to reduce the LCCs would make the project unacceptable.

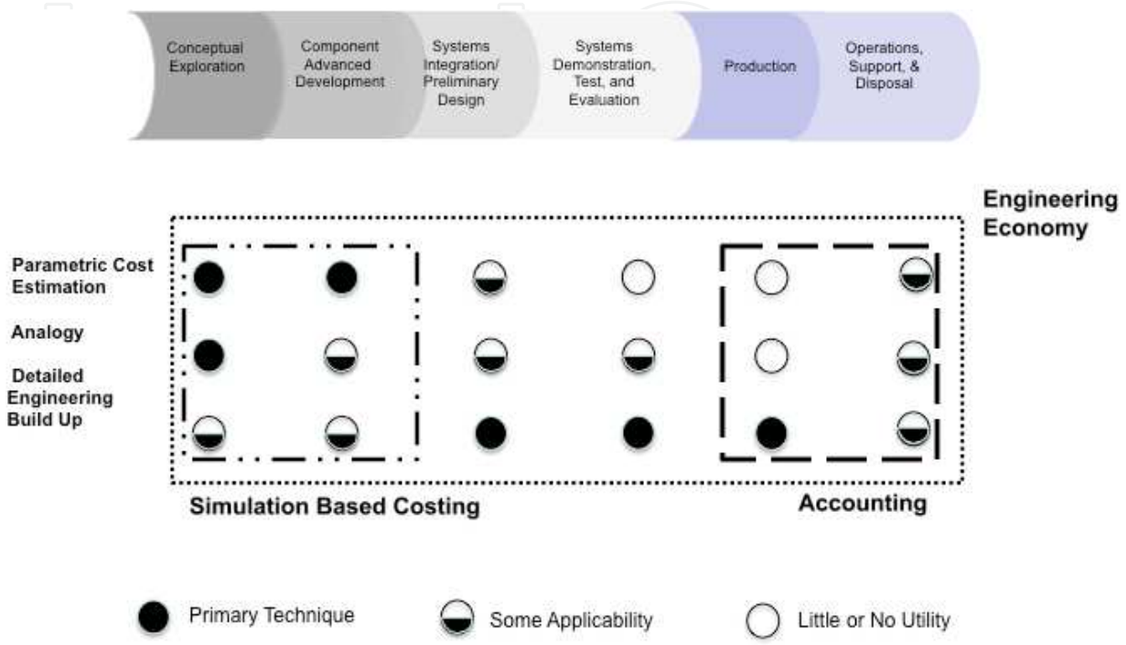

Fig. 1. Cost estimation techniques throughout the life cycle (modified from NASA, 2008)

Technique	Description	Advantages	Disadvantages
Actual Costs/ Extrapolation	Use costs spent during prototyping, hardware engineering development models and early production items to project future costs for the identical system	• Could provide detailed estimate • Reliance of actual development data	• Development data may not reflect cost correctly • Higher uncertainty • Often mistakenly use contract prices to substitute for actual cost • Various levels of detail involvement • Require existing actual production data
Analogy/ Comparative/ Case-based Reasoning	Compare available data from similar project previously completed and adjust estimates for the proposed project	• Reliance of historical data • Less complex than other methods • Save time	• Subjective/bias may be involved • Limited to mature technologies • Reliance of single data point • Hard to identify appropriate analog • Software and hardware often do not scale linearly • Not always possible to find programs of similar scope and complexity

Cost Accounting	Formulate based on the expenditures of reliability, maintainability, and decomposed component cost characteristics	• Reliance of detailed data collection	• Accounting ethics (i.e. cook the books) • Post-production phase strongly preferred • Requires of large and complex data collections • Labor intensive
Detailed Engineering Builds/Bottom-Up	Estimate directly at the decomposed component level that leads to a total combined estimate	• Most detailed at the component level through work breakdown structures • Systemic oriented • Highly accurate • High visibility of cost drivers	• Resource-intensive (time and labor) • May overlook system integration costs • Reliance of stable systems architectures and technical knowledge • Highly prone to double-counting • Lacks ability to capture economies of scale
Expert Judgment/ Delphi Method	Use human experts' knowledge and experience via iterative processes and feedbacks to general consent estimates	• Available when there are insufficient data, parametric cost relationships, or unstable system architectures	• Subjective/bias • Detail cost influence/driver may not be identified • Program complexities can make estimates less reliable • Human experience and knowledge required
Parametric/ Cost Estimating Relationship	Use mathematical expressions and historical data to generate cost relationships models via statistical and regression analysis	• Statistical predictors provide information on expected value and confidence of prediction • Less reliance of systems architectures • Less subjective	• Heavy reliance of historical data • Attributes within data may be too complex to understand • Possibly resource intensive (time and labor) • Difficult to collect data and generate correct cost relationships during cost model development • Limited by data and independent variables
Top-Down	Use the overall project characteristics as the base and generate estimates by decomposing into lower level components and life cycle phases.	• Fast and easy deployment • Minimal project detail required • Systemic oriented	• Less accurate than others • Tend to overlook lower level component details or major cost drivers • Limited detail available for justification

Table 1. Summary of LCCs estimating techniques (from Young et al., 2010)

We have an extensive array of economic techniques and tools at our disposal to predict and monitor LCCs and schedules yet overruns are commonplace and in general are the rule and not the exception; especially for large software enabled systems. Figure 2 shows some of the external and internal factors that we must tackle in conducting cost analysis and then must be addressed when managing the program in the most effective manner.

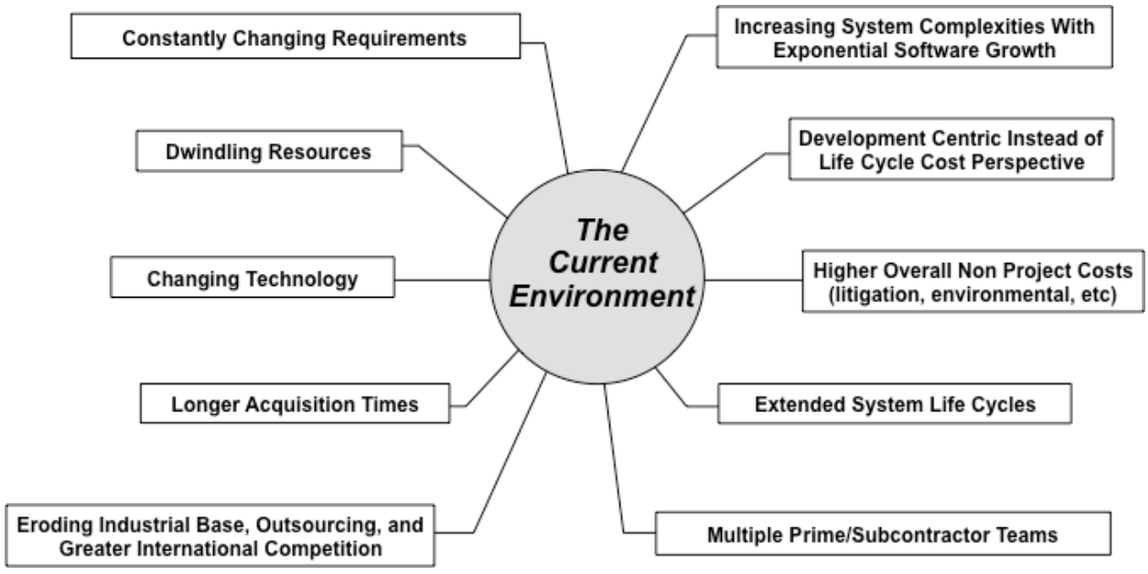

Fig. 2. Some of the factors that can affect the cost of a system (modified from Stevens Institute of Technology, 2008)

The specific purposes utilizing a LCCs perspective in acquisition management, product development, product upgrades, etc., includes:

- Estimate the TOCs to the stakeholder,
- Reduce/capture TOCs through using LCCs tradeoffs in the systems engineering/product development process,
- Control cost through using LCCs contractual provisions in procurements,
- Assist in day-to-day procurement decisions, and
- Understanding TOCs implications to determine whether to proceed to next development phase.

3. Issues surrounding complex systems

Figure 3 shows cost incurred and the ability to influence LCCs over a typical systems life cycle. The figure clearly shows the importance of upfront systems engineering and managing requirements. Because we do not allocate sufficient resources early in a program/project we often make bad engineering decisions that lead to unplanned downstream costs.

From a LCCs perspective what is even more critical is that while developing products and programming and committing funds when we simply do not have the techniques to estimate costs to a high degree of accuracy. The top down tools we used to estimate costs early in the product development cycle are gross rules of thumb at best. When combined

with requirements creep, unstable funding, etc., cost estimates of ± 100% are to be expected. As shown in Figure 4 many factors can contribute to cost and schedule overruns.

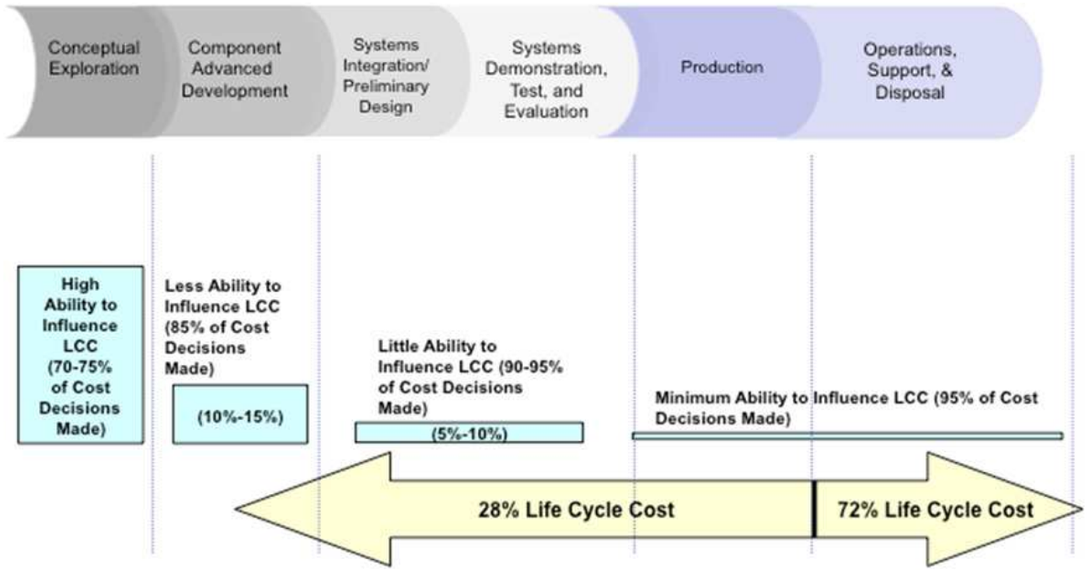

Fig. 3. Costs incurred and committed during our systems life cycle acquisition process (modified from Andrews, 2003)

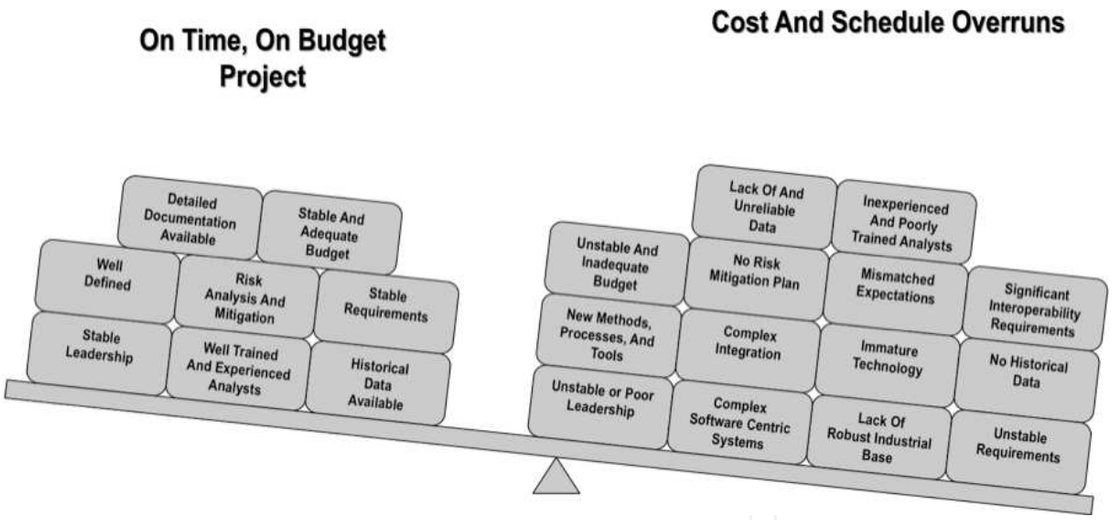

Fig. 4. Challenges cost estimators typically face (modified GAO, 2009)

The techniques for estimating systems costs vary depending upon where we are in the life cycle. Taking our seven-phase model of conceptual exploration, component advanced development; systems integration and preliminary design, systems demonstration and test and evaluation, production, and operations support and disposal, different techniques might be used to estimate costs. For example, early in conceptual exploration the only technique that might be satisfactory is some type of parametric cost estimation techniques such as Constructive Systems Engineering Cost Model (COSYSMO), which will be explained later in detail. As move further into the product development cycle (say at the end of preliminary

design) estimating will be conducted using bottoms up approach/engineering build of the system. Finally, as we enter into production, we will modify our engineering bottoms up model to more accurately reflect the final design elements of hard, software, and interfaces/integration and track costs using formal accounting techniques. Table 2 demonstrates that very early in the product development cycle we simply do not know enough about the system to accurately develop costs. Unfortunately this is when budgets are allocated, bids developed, etc. In order for LCCs to become more accurate we most use software and other formal engineering tools sooner in the design.

Baseline Created	Technical Work Products From Which Estimates Are Developed	Methodologies Used to Develop Cost Estimates
Customer	Customer Requirements • Capabilities • Characteristics Concept of Operations or CONOPS	**Top Down** • Based Upon Number/Complexity of Requirements • Based Upon Number/Complexity of Scenarios • Based Upon Number/Complexity of External Interfaces **Analogous** • Estimates Based Upon Complexity of Technical Work Products Compared to Similar Complexity of Similar Projects *Estimates Are Based On Experience And Historical Data With A ±75% Accuracy*
System	System Requirements Preliminary Architecture	**Top Down** • Based Upon Number/Complexity of Requirements • Based Upon Number/Complexity of Scenarios • Based Upon Technology Maturity • Based Upon Architecture Complexity **Analogous** • Estimate Based on Complexity of Technical Work Products Against Known Projects **Bottom Up** • Estimates Based Upon Architecture *Estimates Are Based On Experience And Formal Design And Systems Engineering (SE) Tools With A ±50% Accuracy*
Component (HW, SW, Process)	Component Requirements • Hardware (HW) and Software (SW) Systems Architecture • Document All HW, SW, Processes, and Interfaces Test Architecture	**Bottoms Up** • Estimates Based Upon Architecture, Technologies Selected, Testing Plan, etc. *Estimates Are Based On Formal Design (Work Breakdown Structure, COCOMO, COSYSMO, Function Point, etc) And SE Tools With A ±10% Accuracy*
Design, Test, and Production	System Into Production HW, SW, and Processes Design and Test Strategy Service Agreements	**Bottoms Up** • Estimates Based Upon Detailed Design, Test Schedules, Implementation Details, and Other Technical Work Products • Delivered Solution Architecture *Estimates Are Detailed Bottoms Up Based Upon All Technical Work Products*

Table 2. Cost and schedule estimates as a function technical baseline work products (modified from Barker, 2008)

4. Hardware, software, systems engineering and management costs

4.1 Hardware costs

If we use a hierarchal approach (a system of systems/enterprise is composed of systems, systems are composed of subsystems, and subsystems are composed of components) any of these levels will be the building block of a bottoms-up estimate. In its simple form, hardware can be separated into physical component that comprise these building blocks plus the labor for estimating purposes. We can think if this as levels of our work breakdown structure or WBS. Note that when developing LCCs for any component of systems is to correctly develop the WBS and assigning hardware (HW), software (SW), integration, etc., for every phase.

As a first cut and if the WBS is developed correctly, we could use these categories as a way to classify costs. Unfortunately, depending upon where you are in the product life cycle we will need to adjust costs to account for technology maturity which might include readiness levels (Technology Readiness Levels or TRLs, Systems Readiness Levels or SRLs, Integration Readiness Levels or IRLs), learning curve issues, etc. NASA (2011) presents a tutorial on TRLs.

As you transition from a top down cost estimating relationship such as COSYSMO, you could use rough relations to estimate these costs over the product life cycle and refine them as the design becomes more final. The WBS and cost models developed must evolve as you move further down the life cycle.

4.2 Software

Software dominates most complex systems. The COnstructive COst Model or COCOMO family of models (see the Center for Systems and Software Engineering, 2011a) are the most widely used software estimation tools in industry. Most developers have validated models for translating lines of code in costs. The challenge for estimating software costs is translating requirements to some type of architecture/requirements to lines of code. Without experience in developing the product software and integrations costs are impossible to develop. The GAO (2009) presents a good overview of the challanges and technqiues for estimating and costing software.

4.3 Interfaces/Integration at the system level

No overarching methodology exists for costing the integration of hardware, software, and developing the interfaces. Interfaces/integration challenges are the key reason why the costs of systems scale non linearly. We know from the DoD, NASA, and other developers of large SoS problems that we do not know how to estimate their costs. The GAO (2009) summarized current major DoD procurements all had experienced significant cost and schedule growth.

4.4 Systems engineering/project management costs

One area that has received significant attention because it is often underfunded and has been connected to major cost overruns is systems engineering and project management (SE/PM). Figure 5 shows some of the SE/PM functions that comprise this category. Stem, et

al (2006) reported that the average SE/PM costs for major aircraft programs had increase from 8% in the 1960s to about 16% in the 1990s of the total development costs. The SE/PM components are significant to controlling costs, schedule, and quality during product design. However, what are the SE/PM concerns post production? These also are significant for upgrades and supportability issues.

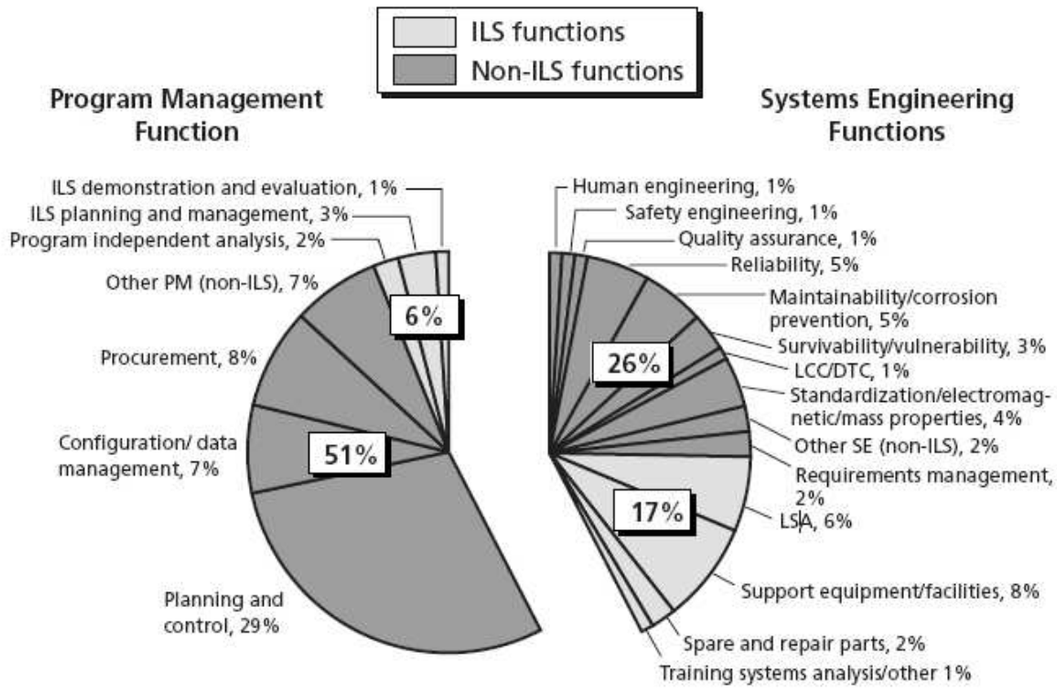

Fig. 5. SE/PM as a function of Integrated Logistics Support (ILS) for a typical Air Force program (from Stem, et al., 2006)

According to Stem et al. (2006) of Rand there is about roughly a 50/50 split of systems engineering and project management costs for most large defense programs. An as shown in Figure 6, these costs can be significant and depending upon maturity, oversight, complexity, etc., can account for about 20% of the development costs. This figure uses lot numbers across product line. Unfortunately, COSYSMO only provides a technique for estimating systems engineering cost during the development phase. Research is underway to identify quantitative means for estimating project management costs from a top down perspective (Young et al, 2011). For services based costing (SBC) to evolve this will be needed.

The COSYSMO is a model that can help people reason about the economic implications of systems engineering on projects. Similar to its predecessor, COCOMO II (Center for Systems and Software Engineering, 2011b), it was developed at the University of Southern California as a research project with the help of BAE Systems, General Dynamics, Lockheed Martin, Northrop Grumman, Raytheon, and SAIC. COSYSMO follows a parametric modeling approach used to estimate the quantity of systems engineering labor, in terms of person months, required for the conceptualization, design, test, and deployment of large-scale software and hardware projects. User objectives include the ability to make Proposal estimates, investment decisions, budget planning, project tracking, tradeoffs, risk

management, strategy planning, and process improvement measurement (see Valerdi, 2005 and 2006).

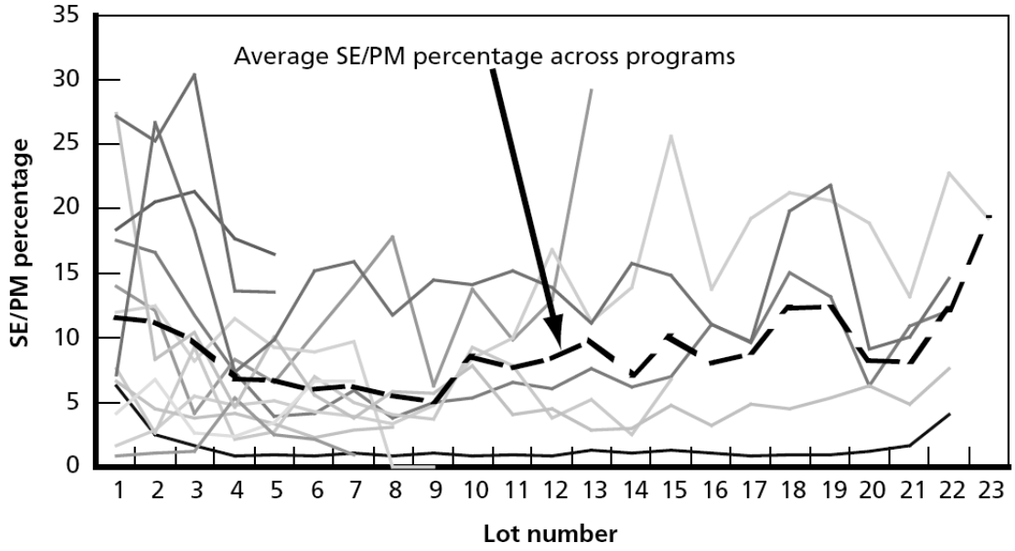

Fig. 6. Average systems engineering and project management costs for 22 major Air Force programs (from Stem et al, 2006)

Each parameter in the COSYSMO Algorithm is part of a Cost Estimating Relationships (CERs) that was defined by systems engineering experts. COSYSMO is typically expressed as (Valerdi, 2005, 2006)

$$\mathrm{PM_{NS}} = A\left(\sum_{k}\left(\omega_{e,k}\Phi_{e,k} + \omega_{n,k}\Phi_{n,k} + \omega_{d,k}\Phi_{d,k}\right)\right)^{E}\prod_{j=1}^{14} EM_j \tag{1}$$

where:

- $\mathrm{PM_{NS}}$ = effort in Person Months (Nominal Schedule)
- A = calibration constant derived from historical project data
- E = represents diseconomies of scale
- k = {REQ, IF, ALG, SCN}
- w_k = weight for "easy", "nominal", or "difficult" size driver
- Φ_k = quantity of "k" size driver
- EM = effort multiplier for the *j*th cost driver. The geometric product results in an overall effort adjustment factor to the nominal effort.

The size of the system is the weighted sum of the system requirements (REQ), system interfaces (IF), algorithms (ALG), and operational scenarios (SCN) parameters and represents the additive part of the model while the EM factor is the product of the 14 effort multipliers.

Obviously there are some shortcomings to this type of approach that would be inherent in any top down model develop early in the life cycle and would include:

- The model is developed on historical data – unless you have significant experience in that domain the model should not be used; and
- Requirements are difficult to use for estimating in that it is difficult to correlate requirements and effort. COSYSMO does recognize this implicitly by distinguishing between pure and equivalent requirements.

5. Methods and tools

5.1 Engineering economy

Engineering economics/economy, is a subset of economics for application to engineering projects. Engineering economics uses relatively simple mathematical techniques to make decisions about capital projects by making comparison of various alternatives. Engineering economy techniques allows for comparisons by accounting for the time value of money. Most engineers are trained in engineering economy and it is the predominate collection of techniques that are used in support of LCCs analysis of complex systems.

Spreadsheets have dramatically changed how we conduct economic analysis of alternatives. What once involved manipulation of equations and tables can now modeled in a spreadsheet using only a few basic commands. The use of spreadsheets are ideal because

- Most problems repetitive calculations that can be expressed as simple formulas as a function of time. Note that Excel® has built in functions for most engineering economy equations.
- Sensitivity analysis is key to conducting good analysis and by properly designing a spreadsheet the parameters can be changed and plots easily developed.
- Complex models can be rapidly and easily built and are for the most part self documenting.
- The user can develop professional reports and plots using the functionality in most spreadsheets.

5.2 Simulation based costing

Systems and enterprises at the most basic level are an integrated composition of elements or sub systems governed by processes that provide a capability to satisfy a stated need or objective. Thus, simulation is an ideal way to analyze these systems. To develop a system or enterprise successfully you must first define the problem that exists, identify the mission requirements (or business drivers) of the organization(s) needing the problem to be solved, evaluate high-level CONOPS for solving the problem, select the concept that makes the most sense in light of the product or mission requirements, develop an operational concept around the selected concept, create architectures and derived requirements for the subsystems, components, and configuration items consistent with the decomposition of the system, design the integration, test and evaluation process for the parts of the system, conduct the integration and test process for the parts of the system, manufacture/assemble the parts of the system, deploy the system, train operators and maintainers, operate/maintain the system, refine the system, and finally retire the system. Simulation can play a key role during each of these phases to assess risk for operational analysis and LCCs. Simulation can be used to prototype the systems, evaluate CONOPS, and used in determining the cost and associated risk.

For simulation based costing (SBC) analysis constructive simulations are the primary analysis tool. Simulation is important for cost analysis because

- the system can be prototyped,
- the only method to model the complex interactions of sub-systems and components,
- conduct CONOPS and "what if" trade space studies, and
- using a combination of the above assess the variability/risk of a LCCs estimate.

Figure 7 demonstrates how simulation can be used throughout the life cycle to assess risk. Note how the distribution of the cost estimate (Y axis) and in the input (triangles on the X axis) both have less variability as the product/project becomes more mature and defined.

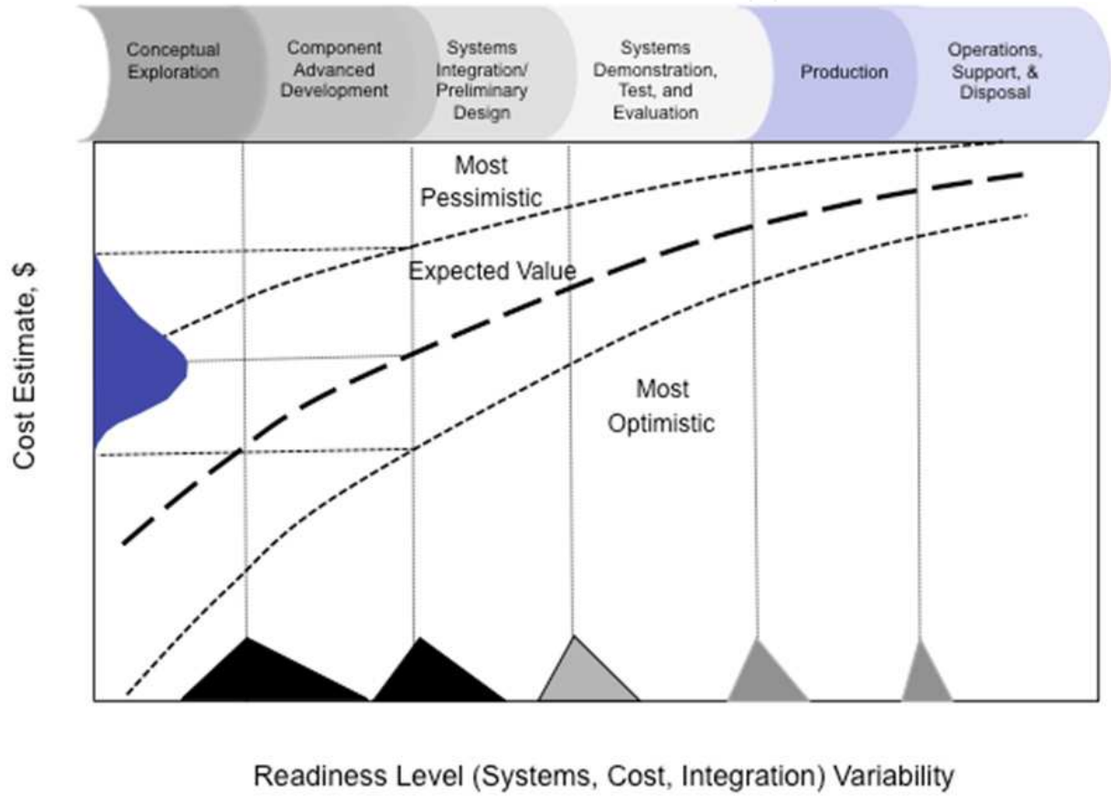

Fig. 7. Cost risk as a function of product life cycle phases

5.3 Parametric cost estimation

The following definitions are used to describe parametric cost estimation (modified from NASA, 2008 and DoD, 1995):

- Parametric Cost Estimates or PCEs - Estimate derived from statistical correlation of historic system costs with performance and/or physical attributes of the system.
- Parametric Cost Model - A mathematical representation of parametric cost estimating relationships that provides a logical and predictable correlation between the physical or functional characteristics of a system, and the resultant cost of the system. A parametric cost model is an estimating system comprising of CERs and other parametric estimating

functions, e.g., cost quantity relationships, inflation factors, staff skills, schedules, etc. Parametric cost models yield product or service costs at designated levels and may provide departmentalized breakdown of generic cost elements. A parametric cost model provides a logical and repeatable relationship between input variables and resultant costs.

- Cost Estimating Relationship or CERs - An algorithm relating the cost of an element to physical or functional characteristics of that cost element or a separate cost element; or relating the cost of one cost element to the cost of another element. CERs can be a functional relationship between one variable and another and may represent a statistical relationship between some well-defined program element and some specific cost, etc. Many costs can be related to other costs or non-cost variables in some fashion but not all such relationships can be turned into CERs.

PCEs utilizes CERs and associated mathematical algorithms, logic, processes to establish cost estimates and are probably the most widely used tool to capture experience. Figure 8 shows a process that can be used for developing CERs for PCEs. Like any mathematical based process, it should only be used for the range described by the "relationship" data.

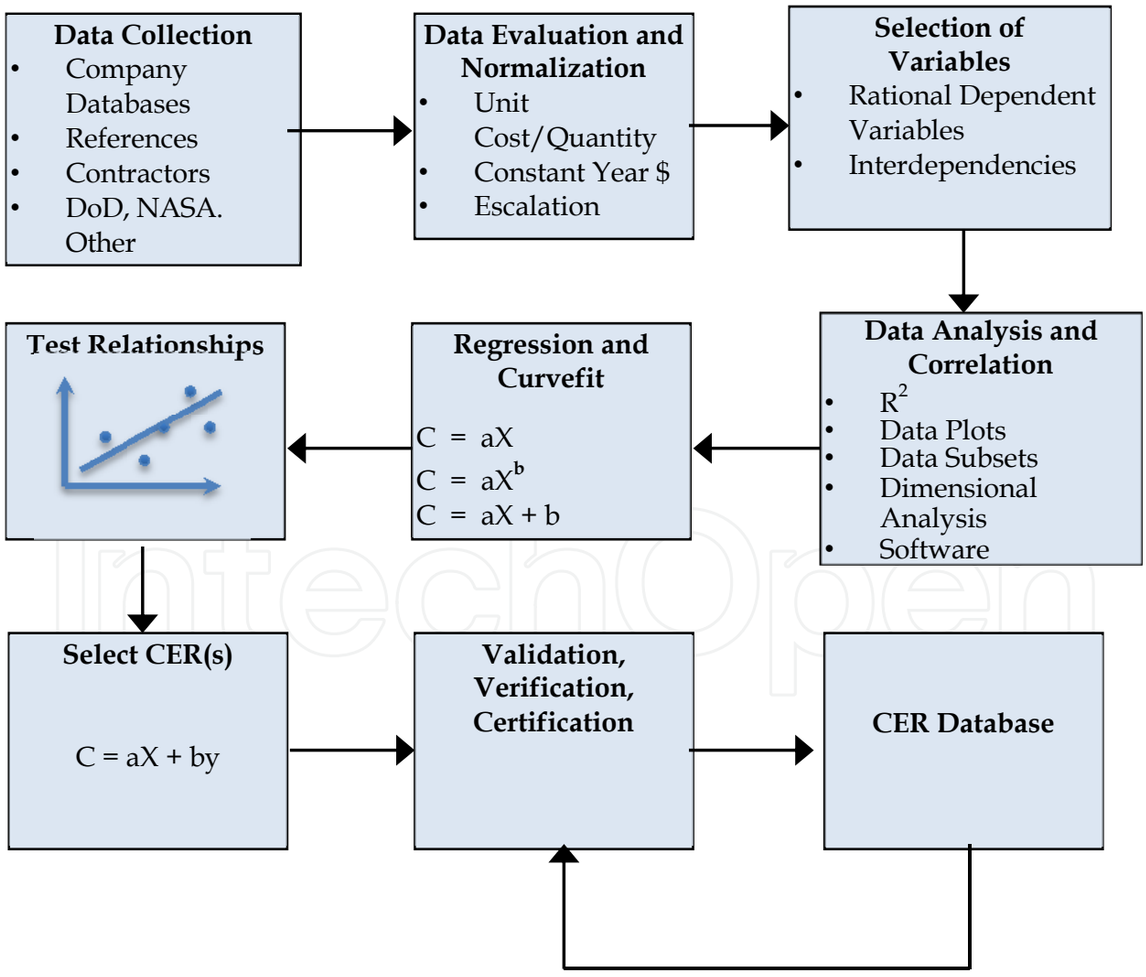

Fig. 8. Process for determining parametric cost estimates (modified from DoD, 1995)

The techniques used in estimating software are much more mature than systems. At best the tools commonly used are estimates and analogies and have little mathematical basis. Whether purely a service's centric or a physical system, most products now have significant software element. The methodology for estimating software has been around for over 30 years and can be classed as PCEs tool. However, because of new languages, hardware/software integration challenges, computer aided software tools, etc., techniques/algorithms must be continually updated. Software estimating is still dominated by experience supplement with quantitative techniques. NASA (2002) has an online handbook describing indepth parametric cost estimating.

5.4 Analogy

Analogy estimates are performed on the basis of comparison and extrapolation using like items or efforts. In many instances this can be accomplished using simple relationships or equations representative of detailed engineering builds of past projects. Obviously, this is the preferred means to conduct a cost estimate based upon past programs that is technically representative of the program to be estimated. Cost data is then subjectively adjusted upward or downward, depending upon whether the subject system is felt to be more or less complex than the analogous program (from NASA, 2008).

5.5 Engineering build or bottom up methodology

The engineering build or bottom up methodology rolls up individual estimates for each element/item/component into the overall cost estimate. This can be accomplished at the WBS element or at the component level. This costing methodology involves the computation of the cost of a WBS element by estimating at the lowest level of detail and computing quantities and levels effort to determine the total system cost. Obviously, this is the most accurate means to develop a cost estimate. The challenge is early in the systems development that a bottom's up approach cannot be utilized because the systems haven't been fully designed. Ideally, you would like to take bottom-up estimates and scale based upon experience. In order to imporve our cost estimates we must conduct bottoms-up estimating soon in the product life cycle. This requires good systems engineering to translate requirements to physical architecture.

6. From requirements to architectures

From a set a system requirements or CONOPs a functional description is developed where the system level requirements or "whats" are translated to "hows" using tools such as functional block diagrams. This functional hierarchy process and interdependencies are shown in Figure 9. The functional description provides the basis for either a physical architecture or a WBS.

7. Costing software

Almost every aspect of our modern society is controlled by software. You can look no further than the defense industry to see how dramatic and persuasive software has become. Consider the following military examples the

- F4 fighter had no digital computer and software (Early 70's),
- F16A fighter had 50 digital processors and 135 thousands of lines of code or KLOC (Late 70's),
- F16D fighter had 300 digital processors and 236 KLOC (Late 80's),
- B-2 bomber has over 200 digital processors and 5,000 KLOC (Late 90's), and
- The US Army's Future Combat Systems (FCS) will have over 16,000 to 50,000 KLOC (Late 00's).

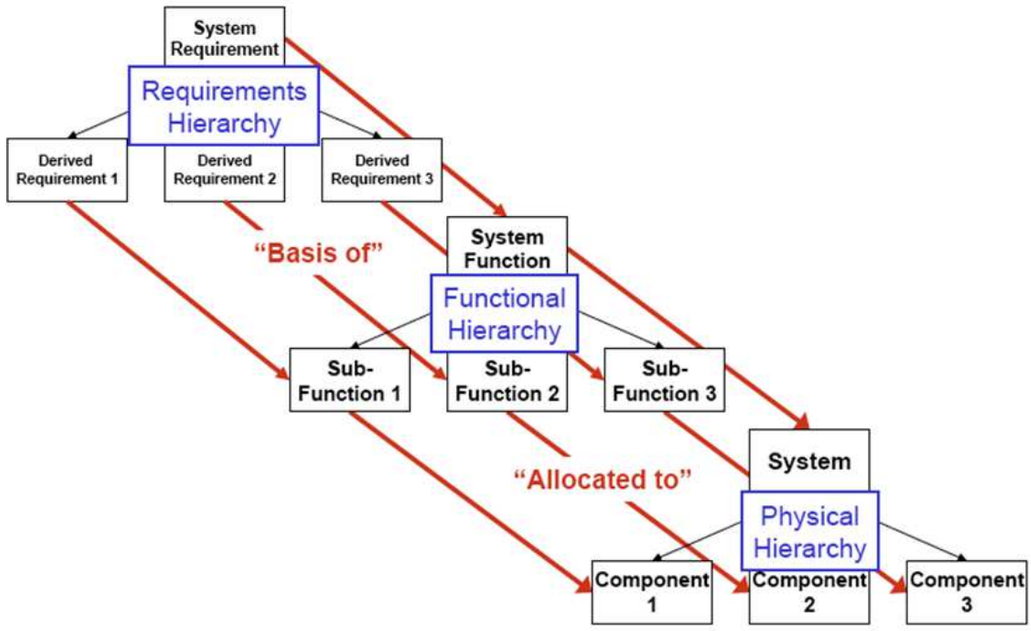

Fig. 9. Role of functional and physical views of a system (from Stevens Institute of Technology, 2009)

Software requirements growth (% of functionality provided by software) has grown from less than 10% in the 1980s to 80% in our current world (National Research Council, 2008).

Software is also redefining the consumer's world. Microprocessors embedded in today's automobiles require software to run, permitting major improvements in their performance, safety, reliability, maintainability, and fuel economy. According to Elektrobit (2007), today's high-end automobiles contain up to 70 electronic control units that control the vehicle's major functions. The average car in 1990 had one million lines of code; by 2010, the average car is expected to have up to 100 million lines of code with software and electronics contributing to over one-third of the cost of a car. New devices in the consumer electronics sector have dramatically changed how we play and manage music and conduct personal computing to extend that we manage our daily activities. As software becomes more deeply embedded in most goods and services, creating reliable and robust software is becoming an even more important challenge. Despite the pervasive use of software, and partly because of its relative immaturity especially with regards to integrating complex hardware and software applications, understanding the economics of software presents an extraordinary challenge.

Engineers typically know how to estimate hardware – we can simply count up the components. However, software and integration/interfaces continue to be the challenge in costing complex systems. Thus, we wrote this chapter to expose readers to the myriad of methods to estimate software. As you will see, historical analysis dominates software cost estimation.

Probably the most important tool in developing a software (or any) cost estimate is to develop some type of functional representation to capture all elements in the life cycle. This includes (modified from DoD, 2005):

A product-oriented family tree composed of hardware, software, services, data, and facilities. The family tree results from systems engineering efforts during the acquisition of a defense materiel item.

A WBS displays and defines the product, or products, to be developed and/or produced. It relates the elements of work to be accomplished to each other and to the end product. A WBS can be expressed down to any level of interest. However the top three levels are as far as any program or contract need go unless the items identified are high cost or high risk. Then, and only then, is it important to take the work breakdown structure to a lower level of definition.

Most models are a mix of expertise based and hybrid because of the subject nature of many of the inputs and algorithms. Expertise is nothing more than subjective human estimating combined with some simple heuristics. One large defense contractor uses the expertise and algorithm to estimate software costs:

- Estimate the number of function points based upon requirements, like projects, etc;
- Use Intermediate or COCOMO II (see Boehm et al, 2000) to estimate the resources required; and then
- Multiple the software development time by 175% to estimate costs.

This is one example of an experienced based algorithm combined with a mathematical model to produce a hybrid technique. Most companies use “rules of thumb” with hybrid techniques to estimate software development costs.

The original COCOMO is an algorithm-based model developed by Boehm (1981) and is used predicts the effort and schedule for a software product development. The model is based on inputs relating to the size of the software and a number of cost drivers that affect productivity COCOMO and drew on a study of about sixty projects with software ranging in size from 2,000 to 100,000 lines of code. Most companies even today use a modified version of one of the COCOMO family of models to estimate software development times and efforts.

The original COCOMO consists of a hierarchy of three increasingly detailed versions (modified from NASA, 2008):

- Basic COCOMO computes software development effort (and cost) as a function of program is good for quick, early, rough order of magnitude estimates of software costs;
- Intermediate COCOMO (Boehm et al, 2000) computes software development effort as function of program size and a set of "cost drivers" that include subjective assessment of product, hardware, personnel and project attributes; and

- Detailed COCOMO incorporates all characteristics of the intermediate version with an assessment of the cost driver's impact on each step (analysis, design, etc.) of the software engineering process engineering.

The basic COCOMO, which is also referred to as COCOMO 81 (Boehm, 1981), is a static model that utilizes a non-linear single valued input equation to compute software development effort (and cost) as a function of software program size. The main input into the model is estimated KDSI. The model takes the form:

$$E = aS^b \quad (2)$$

where: E = effort in person-months,
S = size of the software development in KDSI, and
a, b = values dependent on the development mode

Note that all models that COSYMO and other COCOMO based models all use this type of exponential model. Typically they all follow the form presented in Equation 2 with additional multiplicative factors.

8. Cost management

8.1 Introduction

Engineering cost management can be defined as the process to identify, allocate, and track resources needed to meet the stakeholder's requirements. An integrated, process-centered, all backed with quantifiable data and documented processes provides real and tangible benefits to all stakeholders. Engineering cost management can best be described as an integrated, process-centered, measurable, and disciplined approach to LCCs and management to make the tradeoffs between cost, performance, schedule, and risk. Good cost management practices, supported by sound analysis, can lead to (modified from NASA, 2008):

- Complete, unambiguous, and documented functional requirements in order to meet LCCs goals;
- Bounded and clearly defined product functional expectations and acceptance criteria, understood and agreed to by all stakeholders;
- More accurate, credible, and defensible scope, cost, and schedule estimates with realistic assessments of risk;
- More complete and timely risk identification, leading to more effective risk mitigation;
- A basis for properly quantifying, evaluating, and controlling the acceptance and timing of changes to requirements (i.e., precluding "scope creep");
- Final products that deliver better reliability, adaptability, usability, performance, maintainability, supportability, and functionality -- in short, higher quality and value;
- Insight into near, mid and long term technology, design, infrastructure and operational investment needs as they relate to different effects on the phases and trade-offs within the life-cycle;
- Earlier and more consistent visibility to problems (fewer surprises);
- Understanding the costs for each step in the development process;
- More efficient project management; and
- Organizational credibility and reputation.

Engineers play a critical role in corporate or business planning. Engineers are involved in cost management from top-level corporate planning to costing components and sub systems. All require the same basic understanding of time value of money, risk, and life cycle perspective.

Engineering cost management is employed as a means of balancing a project's scope and expectations of risk, quality, and technical performance to ensure that the most cost effective solution is delivered and consists of three steps:

1. Define the requirements, level of quality desired, and the budget,
2. Ensure that the risk, scope, and quality are aligned with the budget, and
3. Monitor and manage the balance of these four components throughout the life of the project by using sound engineering techniques.

The ability to use analysis techniques such as those discussed allow an engineer to conduct defendable and rigorous analysis that can not only provide representative costs but can help scope a technical problem.

One important technique to help manage costs is cost as an independent variable (CAIV). Though mainly a technique that is used solely by government, its underlying principles have utility in the commercial sector. The challenges of managing the costs of open source and off the shelve technology presents a unique costing challenge because integration not development is the key cost driver. The complexity, especially given the amount of software in most modern systems, Lastly, formal tracking using project management techniques to estimate, track, and manage costs. This is beyond the scope of this chapter but is an important for managing costs and are commonly used.

8.2 Cost as an independent variable

Cost as an Independent Variable (CAIV) is a formal methodology for reducing TOCs while maintaining performance and schedule objectives. It involves developing, setting, and refining cost objectives in a systematic method while meeting owner/user requirements. CAIV entails setting aggressive, realistic cost objectives for acquiring systems and managing program risks to obtain those objectives. Cost objectives must balance against market and budget realties with projected out-year resources, taking into account existing technologies as well as the high-confidence matriculation of new technologies (from Kaye, et al, 2000). In essence the CAIV concept means that, once the system performance and objective costs are decided (on the basis of cost-performance trade-offs), then the acquisition process will make cost more of a constraint, and less of a variable, while obtaining the needed capability of the system. Figure 10 shows this graphically.

CAIV is founded upon two primary principles. First, LCCs are constrained. Unfortunately, this is all to often limited to development and production costs. Whereas some programs do obtain additional funding when needed, such funding is often at the expense of other business units, programs, or future modernization. Second, “trade space” is the foundation for smart decisions. Trade space is the range of alternatives available to the buyers. It is four-dimensional, comprising performance, TOCs, schedule, and risk impacts (from Kaye, et al., 2000). Many of the methods presented such as SBC can be used for this trade space analysis.

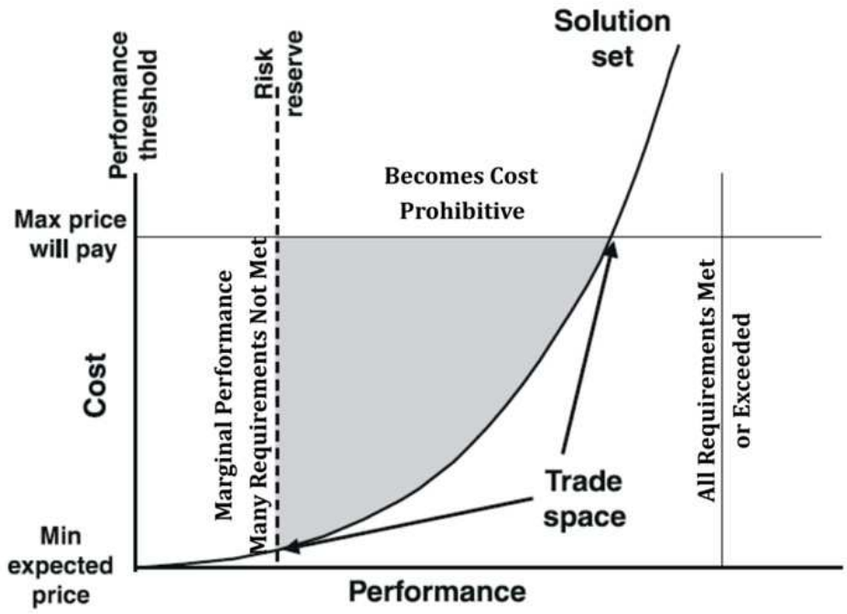

Fig. 10. CAIV representation (modified from Kaye, et al., 2000)

8.3 Formal cost accounting

Cost accounting is obviously the best way to track costs. The emergence of activity based costing techniques have made the engineers job easier when trying to ascertain true costs. Activity Based Costing (ABC) tracks costs—both direct and indirect—to their source. While traditional accounting practices have concentrated on evaluating inventory for asset based reporting, ABC links the resources consumed to the activities performed and then links these activities directly to their products. As a result, ABC provides a basis for strategic product and service pricing by capturing the direct relationships between costs, activities, and products. This is particularly useful when the primary cost factors are directly traceable to individual products or traditional direct costs. Most costs in industrial companies today are indirect, resulting, when indirect costs are uniformly allocated across products, in invalid management support information. This is particularly true in a service organization—commercial or government—attempt to use traditional inventory accounting techniques for management support will inevitably lead to inappropriate decisions.

All engineers need to understand the basics of cost accounting. As systems become more complex, the role of the engineer has diminished in terms of developing detail proposals. Most engineers now develop LCCs estimates for the system. Unless you are working at the senior management level, you do not need an in depth accounting background.

9. Summary

Costing systems is complex and consists of a variety of techniques to include analogies, PCEs, and detailed bottom-ups modeling. Unlike the mature knowledge encompassed by the traditional engineering disciplines, the techniques and tools for costing and managing complex systems are rapidly evolving and being driven mainly by the commercial sector. Also, the MPTs and techniques are often not presented in the open literature because of the

competitive advantage afforded any company that can accurately estimate the LCCs of a product. Thus much of the MPTs presented were gleamed from government sources especially the DoD the National Aeronautical and Space Administration. Fortunately, the DoD and NASA are in many ways the intellectual thought leader on costing and estimating of complex systems because of the sheer size and complexity of their projects/programs. There is probably no one size fits or collect of MPTs, and certainty no substitution for experience, that are repeatable for LCCs estimation. However, much research, especially for techniques applicable early in the life cycle, is needed to better ascertain true LCCs.

Good engineers follow a disciplined and structured approach when developing a product/system. Costing hardware, software, and integration requires an understanding of many MPTs and terminology that few engineers have received formal training. Once technical characteristics have been ascertained from the requirements, selecting the right MPTs is critical to accurately determining costs early in the development cycle and estimating realistic LCCs.

In the evaluation and reengineering of existing systems, the functional analysis serves as a basis for developing WBS or CBS leading to the collection of costs by functional area. Unfortunately, if you can develop architectures/WBS you have a well-understood system suitable for realistic costs estimates which is often long after a budget has been establish.

10. References

Andrews, Richard, "An Overview of Acquisition Logistics," Fort Belvoir, VA: Defense Acquisition University, 2003, available online at https://acc.dau.mil/CommunityBrowser.aspx?id=32720, accessed on April 2, 2007

Barker, Bruce, personal note, April, 2008

Boehm, Barry, Software Engineering Economics, Prentice-Hall, 1981

Boehm, Barry, Abts, Chris, A. Brown, Winsor, Chulani, Sunita, Clark, Bradford K., Horowitz, Ellis, Madachy, Ray, Reifer, Donald J. and Steece, Bert, Software Cost Estimation with COCOMO II, Englewood Prentice-Hall, 2000

Center for Systems and Software Engineering, COCOMO Tools, University of Southern California, 2011, available onine at http://csse.usc.edu/csse/tools/, accessed on 19 August 2011a

Center for Systems and Software Engineering, COCOMO II, University of Southern California, 2011, available online at http://sunset.usc.edu/csse/research/COCOMOII/cocomo_main.html, accessed on 19 August 2011b

Department of Defense, "Parametric Cost Estimating Handbook," Joint Government/Industry Initiative, Fall, 1995

Department of Defense Handbook, "Work Breakdown Structure for Defense Material Items," available onine at http://www.acq.osd.mil/pm/currentpolicy/wbs/MIL_HDBK-881A/MILHDBK881A/WebHelp3/MIL-HDBK-881A%20FOR%20PUBLICATION%20FINAL% 2009AUG05.pdf, accessed on 30 July, 2005

Elektrobit, "Freescale and Elektrobit IntroduceAutosar Sfotware Bundle for Automotive Microcontrollers," available onine at http://www.elektrobit.com/news-987-195-freescale_and_elektrobit_introduce_autosar_software_bundle_for_automotive_microcontrollers, 10 October 2007, accessed on 19 August 2011

Government Accounting Office (GAO), "Cost Estimating and Assessment Guide Best Practices for Developing and Managing Capital Program Costs," GAO-09-3SP, March 2009, available online at http://www.gao.gov/new.items/d093sp.pdf, accessed 19 August 2011

IBM, "Software Estimation, Enterprise-Wide, Part I: Reasons and Means," available onine at http://www.ibm.com/developerworks/rational/library/jun07/temnenco/index.html, accessed November 18, 2008

Kaye, M. A., Sobota, M. S., Graham, D. R., and Gotwald, A. L., 2000, "Cost as an Independent Variable: Principles and Implementation," Available onine at http://www.dau.mil/pubs/arq/2000arq/kaye.pdf, from the Acquisition Review Quarterly, Fall, 2000, accessed January 2008

National Aeronautics Space Administration, "Cost Estimating Handbook," Available onine at www.nasa.gov/ceh_2008/2008.htm, accessed 28 January 2010

National Aeronautics Space Administration, "Parametric Cost Estimating Handbook," Available onine at http://cost.jsc.nasa.gov/PCEHHTML/pceh.htm, 2009, accessed 19 August 2011

National Aeronautics Space Administration, Technology Readiness Levels Demystified , Available online at http://www.nasa.gov/topics/aeronautics/features/trl_demystified.html, 20 August 2010, accessed 19 August 2011

National Research Council of the National Academies, Air For"Pre-Milestone A Systems Engineering - A Retrospective Review and Benefits for Future Air Force Systems Acquisition," Air Force Studies Board, 2008

Stem, David E., Boito, Michael and Younossi, Obaid, "Systems Engineering and Program Management - Trends and Costs for Aircraft and Guided Weapons Programs," ISBN 0-8330-3872-9, Published by the RAND Corporation, Santa Monica, CA, 2006

Stevens Institute of Technology, "SYS 625 Fundamentals of Systems Engineering Class Notes," 2008

Stevens Institute of Technology, "SYS 650 System Architecture and Design," Course Notes, 2009

Valerdi, Ricardo, "The Constructive Systems Engineering Costing Model (COSYSMO)," A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy, University of Southern California, August 2005, Available onine at http://csse.usc.edu/csse/TECHRPTS/PhD_Dissertations/files/Valerdi_Dissertation.pdf, accessed 19 August 2011

Valerdi, Ricardo, "Academic COSYSMO User Manual - A Practical Guide for Industry and Government," Version 1.1, MIT Lean Aerospace Initiative, September 2006

Young, Leone Z., Farr, John V., Valerdi, Ricardo, and Kwak, Young Hoon, "A Framework for Evaluating Life Cycle Project Management Costs on Systems Centric Projects,"31th Annual American Society of Engineering Management Conference, Rogers, AK, October, 2010

Young, Leone Z., Wade, Jon, Farr, John V., Valerdi, Ricardo, and Kwak, Young Hoon, "An Approach to Estimate the Life Cycle Cost and Effort of Project Management for Systems Centric Projects," International Society of Parametric Analysts (ISPA) and the Society of Cost Estimating and Analysis (SCEA)2011 ISPA/SCEA Joint Annual Conference and Training Workshop, Albuquerque, New Mexico, June 7 - 10, 2011

Systems Engineering - Practice and Theory
Edited by Prof. Boris Cogan

ISBN 978-953-51-0322-6
Hard cover, 354 pages
Publisher InTech
Published online 16, March, 2012
Published in print edition March, 2012

The book "Systems Engineering: Practice and Theory" is a collection of articles written by developers and researches from all around the globe. Mostly they present methodologies for separate Systems Engineering processes; others consider issues of adjacent knowledge areas and sub-areas that significantly contribute to systems development, operation, and maintenance. Case studies include aircraft, spacecrafts, and space systems development, post-analysis of data collected during operation of large systems etc. Important issues related to "bottlenecks" of Systems Engineering, such as complexity, reliability, and safety of different kinds of systems, creation, operation and maintenance of services, system-human communication, and management tasks done during system projects are addressed in the collection. This book is for people who are interested in the modern state of the Systems Engineering knowledge area and for systems engineers involved in different activities of the area. Some articles may be a valuable source for university lecturers and students; most of case studies can be directly used in Systems Engineering courses as illustrative materials.

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen